Agile:

1.

a. As a vanilla git power-user that has never seen GiggleGit before, I want to quickly understand how GiggleGit works and how it is different from regular Git. This is so that I can determine whether it will fit into my workflow and if so integrate into my projects without a steep learning curve.
b. As a team lead onboarding an experienced GiggleGit user, I want to ensure that I can set up and configure GiggleGit for the collaborative projects that my team and I work on. Since teams use more advanced tools on git than individual users, making sure GiggleGit has the same functionality in terms of collaboration is important.

2.

As a new user onboarding onto GiggleGit, I want a clear, guided setup process that helps me configure my environment and preferences, so I can start using the system with minimal issues and feel confident in my setup.

Task: Create a guided and interactive setup process for new users.

Tickets:

1. Design user onboarding flow
   Design a step-by-step user flow that guides new users through configuring their environment, selecting preferences, and linking their Git repositories. Make sure it is easy to follow and covers all necessary steps for getting started.
2. Implement guided setup in the GiggleGit UI
   Build the designed onboarding flow into the GiggleGit interface. This should include prompts, input fields, and tooltips to walk users through setting up their environment, preferences, and linking repositories to start using GiggleGit.

3.

This is not a user story as it lacks specificity and does not have the elements of a user story such as context. The sentence does not describe what kind of user this is and why they are seeking to authenticate on a new machine. Another factor that is not included is the reason why the user needs to do this; by knowing the motivation behind this, developers can easily understand the issue at hand.

Formal Requirements:

1.

Goal: Ensure that users can easily differentiate between regular merges and SnickerSync merges, and make sure that SnickerSync does not reduce efficiency but rather adds an element of humor to work.

Non-goal: Redesign the entire Git/GiggleGit interface around SnickerSync merges. The focus is on integrating SnickerSync into existing workflows.

2.

Non-functional requirements:

1. PM Access Control
    a. PMs must have the ability to create, edit, and remove aspects within the system to update/experiment with new versions of SnickerSync.
2. User Study Randomization
    a. The system must randomly assign users into control/variant groups during the user study; this will ensure unbiased distribution for testing the “snicker” feature.

3.

Functional requirements for #1:

1. Develop an admin interface that allows PMs to create and edit different snickering behaviors.
2. Implement access controls so that only PMs have the ability to modify, save, or delete snickering concepts, ensuring these features are protected from regular user access.

Functional requirement for #2:

1. Implement an algorithm that automatically assigns users to either the control group or the variant group upon logging into the study.
2. Ensure that user assignments persist across sessions(this means that once a user is assigned to a group, they will remain in that group throughout the entire study.

5. Dependencies